METHOD AND SYSTEM FOR GENERATING AND SEARCHING AN OPTIMAL MAXIMUM LIKELIHOOD DECISION TREE FOR HIDDEN MARKOV MODEL (HMM) BASED SPEECH RECOGNITION

FIELD OF THE INVENTION

The present invention relates generally to speech processing and to large vocabulary continuous speech recognition (LVCSR) systems. More particularly, the present invention relates to a method and system for generating and searching an optimal maximum likelihood decision tree (OML-DCSTree) for hidden markov model (HMM) based speech recognition.

BACKGROUND OF THE INVENTION

A speech recognition system recognizes a collection of spoken words ("speech") into recognized phrases or sentences. A spoken word typically includes one or more phones or phonemes, which are distinct sounds of a spoken word. Thus, to recognize speech, a speech recognition system must determine relationships between the words in the speech. A common way of determining relationships between words in recognizing speech is by using state decision tree clustering under a continuous density hidden markov model (HMM).

Typically, a HMM uses a series of transitions from state to state to model a letter, a word, or a sentence. Each state can be referred to a phone or a triphone (phone having a left and right context). Each arc of the transitions has an associated probability, which gives the probability of the transition from one state to the next at an end of an observation frame. As such, an unknown speech signal can be represented by ordered states with a given probability. Moreover, words in an unknown speech signal can be recognized by using the ordered states of the HMM.

Currently, there are two popular approaches in building a decision tree-based HMM, which are a rule-based decision tree approach and a data driven approach. The rule-based decision tree approach uses a set of simple or composite phonetic context questions (questions requiring simple "yes" or "no" answers) as a set of rules to construct a HMM state decision tree according to a maximum likelihood (ML) principle. By way of contrast, the data driven approach groups triphones using specific training data obtained during a training process.

A disadvantage of the rule-based decision tree approach is that there is no fairness guaranty for each individual rule regardless if it is derived by a linguistic expert, a database, or by both. That is, for simple phonetic context question based rules, all triphones are grouped into two yes/no classes according to their phonetic contextual information. However, some of the triphones, contrary to the rule, may have a better maximum likelihood (ML) criteria in another class. The data driven approach can provide better guarantees for optimal ML classification for known triphones (i.e., triphones that are obtained during the training process). A disadvantage of the data driven approach, however, is that the tree built with the data driven approach has problems dealing with triphones that did not appear during the training process, but do appear in the recognition process. Consequently, unknown triphones, which were not detected during the training process, may lower speech recognition accuracy.

BRIEF DESCRIPTION OF THE DRAWINGS

The features and advantages of the present invention are illustrated by way of example and not intended to be limited by the figures of the accompanying drawings in which like references indicate similar elements, and in which:

FIG. 1 is a diagram illustrating an exemplary digital processing system for implementing the present invention;

FIG. 2 is a block diagram of a speech processing system according to one embodiment;

FIG. 3 is a functional flow diagram illustrating an operation using an optimal maximum likelihood decision tree according to one embodiment;

FIG. 4 is a flow chart illustrating an operation to build an optimal maximum likelihood decision tree according to one embodiment;

FIG. 5 is an exemplary diagram of tree nodes in a decision tree;

FIG. 6 is an exemplary diagram of a maximum likelihood decision tree to illustrate decoding using the maximum likelihood decision tree; and

FIG. 7 is a flow chart illustrating an operation to decode speech using an maximum likelihood tree according to one embodiment.

DETAILED DESCRIPTION

According to one aspect of the present invention a method is disclosed for generating and searching an optimal likelihood decision tree (OML-DSCTree) for hidden markov model (HMM) based speech recognition. For one embodiment, speech signals are received. The received speech signals are processed to generate a plurality of phoneme clusters. The phoneme clusters (e.g., biphone or triphone clusters) are grouped into a first cluster node and a second cluster node according to their answers to phonetic context questions. A determination is made if a phoneme cluster in the first cluster node is to be moved into the second cluster node based on a likelihood increase of the phone cluster of the first cluster node from being in the first cluster node to being in the second cluster node.

The following speech processing techniques described herein can provide higher speech recognition accuracy by reshuffling phoneme clusters between first node cluster and a second node cluster based on the likelihood increase. The following speech processing techniques can be used to build a decision tree, which can ensure optimal maximum likelihood (ML). By controlling a threshold, the following speech processing techniques can be used to generate a decision tree with any combination of a rule-based approach and a data driven approach under a unified decision tree structure based on ML principles.

In the following description, an optimal maximum likelihood decision tree (OML-DCSTree) is described for speech recognition, however, the OML-DCSTree can also be used for pattern recognition applications.

FIG. 1 is a diagram illustrating an exemplary digital processing system 100 for implementing the present invention. The speech processing and speech recognition techniques described herein can be implemented and utilized within digital processing system 100, which can represent a general purpose computer, portable computer, hand-held electronic device, or other like device. The components of digital processing system 100 are exemplary in which one or more components can be omitted or added. For example, one or more memory devices can be utilized for digital processing system 100.

Referring to **FIG. 1**, digital processing system 100 includes a central processing unit 102 and a signal processor 103 coupled to a display circuit 105, main memory 104, static memory 106, and mass storage device 107 via bus 101. Digital processing system

100 can also be coupled to a display 121, keypad input 122, cursor control 123, hard copy device 124, input/output (I/O) devices 125, and audio/speech device 126 via bus 101.

Bus 101 is a standard system bus for communicating information and signals. CPU 102 and signal processor 103 are processing units for digital processing system 100. CPU 102 or signal processor 103 or both can be used to process information and/or signals for digital processing system 100. Signal processor 103 can be used to process speech or audio information and signals for speech processing and recognition. Alternatively, CPU 102 can be used to process speech or audio information and signals for speech processing or recognition. CPU 102 includes a control unit 131, an arithmetic logic unit (ALU) 132, and several registers 133, which are used to process information and signals. Signal processor 103 can also include similar components as CPU 102.

Main memory 104 can be, e.g., a random access memory (RAM) or some other dynamic storage device, for storing information or instructions (program code), which are used by CPU 102 or signal processor 103. For example, main memory 104 may store speech or audio information and instructions to be executed by signal processor 103 to process the speech or audio information. Main memory 104 may also store temporary variables or other intermediate information during execution of instructions by CPU 102 or signal processor 103. Static memory 106, can be, e.g., a read only memory (ROM) and/or other static storage devices, for storing information or instructions, which can also be used by CPU 102 or signal processor 103. Mass storage device 107 can be, e.g., a hard or floppy disk drive or optical disk drive, for storing information or instructions for digital processing system 100.

Display 121 can be, e.g., a cathode ray tube (CRT) or liquid crystal display (LCD). Display device 121 displays information or graphics to a user. Digital processing system 101 can interface with display 121 via display circuit 105. Keypad input 122 is a alphanumeric input device for communicating information and command selections to digital processing system 100. Cursor control 123 can be, e.g., a mouse, a trackball, or cursor direction keys, for controlling movement of an object on display 121. Hard copy device 124 can be, e.g., a laser printer, for printing information on paper, film, or some other like medium. A number of input/output devices 125 can be coupled to digital processing system 100. For example, a speaker can be coupled to digital

processing system 100. Audio/speech device 126 can be, e.g., a microphone with an analog to digital converter, for capturing sounds of speech in an analog form and transforming the sounds into digital form, which can be used by signal processor 203 and/or CPU 102, for speech processing or recognition.

The speech processing and speech recognition techniques described herein can be implemented by hardware and/or software contained within digital processing system 100. For example, CPU 102 or signal processor can execute code or instructions stored in a machine-readable medium, e.g., main memory 104, to process or to recognize speech.

The machine-readable medium may include a mechanism that provides (i.e., stores and/or transmits) information in a form readable by a machine such as computer or digital processing device. For example, a machine-readable medium may include a read only memory (ROM), random access memory (RAM), magnetic disk storage media, optical storage media, flash memory devices. The code or instructions can be represented by carrier wave signals, infrared signals, digital signals, and by other like signals.

FIG. 2 is a block diagram of a speech processing system 200 according to one embodiment. The speech recognition system can be implemented, e.g., in digital processing system 100 as described in **FIG. 1**. Referring to **FIG. 2**, block diagram 200 includes an audio/speech device 204 for receiving and processing speech signals 202 and a signal processor 206 for processing speech signals 402 from audio/speech device 204. For one embodiment, signal processor 206 can build an OML-DCSTree as described in **FIG. 4** to be stored as acoustical models 208 in a training process. For another embodiment, signal processor 206 can use acoustical models 208 (e.g., an OML-DCSTree) to recognize speech during a speech recognition process.

Audio/speech device 204 is an audio and speech receiving mechanism. For example, audio/speech device 204 can be a microphone. A user can speak into audio/speech device 203 in which acoustics (i.e., analog signals) are provided to audio/speech device 203. Audio/speech device 204 can convert the analog signals (i.e., speech signals 202) into digital form.

Signal processor 206 is a processing device, which can be used during a training process or a continuous speech recognition process. During a training process, signal processor 206 builds relationships for the words contained in speech signals 202 to

create acoustical models 208. For example, acoustical models 208 can be based on a hidden markov model (HMM). During a speech recognition process, signal processor 206 receives unknown speech signals 202. Signal processor searches the acoustical models 208 to match words in the unknown speech signals with known states in the acoustical models 208. If words in the unknown speech signals match known states of the acoustical models 208, processor 206 can output a recognized output such as a text file including known words, phrases, or sentences.

FIG. 3 is a functional flow diagram illustrating an operation 300 using an optimal maximum decision likelihood (OML-DCSTree) according to one embodiment. Referring to **FIG. 3**, at functional block 302, speech is inputted into a feature extractor. The feature extractor extracts data from the analog form of the speech. For example, the feature extractor can extract frequency data such as, for example, mal-frequency cepstrum coefficients (MFCC) and its first and second derivatives from the speech.

At functional block 304, the data from the feature extractor can be either trained (for a training process) or recognized (for a recognition process). At functional blocks, 306 and 308, if data from the feature extractor is to be trained, the data is trained into an OML-DCSTree based on a HMM. For example, the operation as explained in **FIG. 4** can be used to build the OML-DCSTree.

At functional block 310, if data from the feature extractor is to be recognized, the data can be recognized using an OML-DCSTree HMM recognizer, which can use the OML-DCSTree based HMM or a pre-trained language model (LM) to generate recognized speech. For example, the operation as explained in **FIG. 7** can be used to recognize speech using the OML-DCSTree. The LM component of the system depends on the task, which can be a popular N-gram model for a dictation-like system task or a finite-state-grammar model for a dialog task.

FIG. 4 is a flow chart illustrating an operation 400 to build an optimal maximum likelihood decision tree (OML-DCSTree) according to one embodiment. The following operation 400 is built on top of a phonetic question based state decision tree framework such as a rule-based approach framework. That is, for each question in a question set, all the triphones are first classified into two (yes/no) nodes. For purposes of explanation, operation 400 starts at operation 402.

Referring to **FIG. 4**, at operation 402, all the triphone clusters are collected that are to be clustered starting from a root node. At operation 404, cluster all the triphone

clusters into two children nodes: Y-Node and N-Node according to their Y/N answers to the "best" question.

At operation 406, a node is arbitrarily picked, e.g., a Y-Node and the operation is to find one cluster (Y_B) that gives most likelihood increase (L_y) if it is in the N-Node. At operation 407, a Y-Node to a N-Node class member transferring is noted.

At operation 408, a determination is made if $L_y > T_O$; T_O being a threshold value. If L_y is greater than T_O at operation 410, the cluster Y_b is moved into its sister N-Node and saved on the N-Node. If L_y is not greater than T_O at operation 412, the operation is to find one cluster (N_b) that gives most likelihood increase (L_N) if it is in the Y-Node. At operation 414, a N-Node to a Y-Node class member transferring is noted.

At operation 416, a determination is made if $L_{N_1} > T_0$; T_0 being a threshold value. If L_{N_1} is greater than T_0, at operation 418, the cluster N_b is moved into the Y-Node and saved in the Y-Node and operation continues back to operation 406. If N_1 is not greater than T_0, then operation 400 continues at operation 420.

At operation 420, a determination is made if $L_y < T_1$ & $L_N < T_1$. If L_y is not less than T_1 or L_N is not less than T_1, operation 400 continues back to operation 406. If L_y is less than T_1 & L_N is less than T_1, then operation 400 continues at operation 422.

At operation 422, at determination is made if a certain depth is reached for the decision tree. That is, the above operation is repeated at all levels of the decision tree until a certain dept of the decision tree is reached. If a certain depth is not reached, operation 400 finds the next best node to split and continues back at operation 404. If a certain depth is reached, operation 400 ends.

Pseudo-code for building an OML-DCSTree

The following is exemplary pseudo-code for building an OML-DCSTree as described above with respect to **FIG. 4**. The pseudo-code can be implemented using known programming languages such as "C" or "C++."

Let $\mathbf{N} = \{n1, n2, ..., n_M{}^N\}$ be the node corresponding the "no-answer" for the best question Q_b, $\mathbf{Y} = \{y1, y2, ..., y_M{}^Y\}$ the node corresponding the "yes-answer" to the best question at its parent node, and $M = M^N + M^Y$ be the total number of seen triphones at their parent node.

1 Initiate conventional decision tree (omitted).

2 Reshuffle the class members between Y/N-nodes based on the likelihood increase for Question Q_b.

```
float prob_old = prob (N) + Prob (Y);
float prob_best = prob_old;

For(i=1, i<My;i++){
        float prob_minus = prob (N+i);
        float prob_plus = prob (Y-i);

        if( (prob_minus+prob_plus) > prob_best ) {
        index=i;
        prob_best=prob_minus+prob_plus;
        }
}

if( (prob_best - prob_old)>Certain criterion ){
        Move the triphone cluster indexed by "index" cluster to N-node
from the Y-node
} else {
 break;
}
```

The above procedure will repeated starting from N-Node;
Then the above two-way class membership reshuffle is repeated until it meets some condition.

3 Record the new class members for the two new Y/N-nodes.

4 Repeat the procedure 2-3 until it reach certain depth of the DCSTree.

The decision tree can be trained in the conventional way.

FIG. 5 is an exemplary diagram of tree nodes in a decision tree 500. The decision tree 500 includes a root node 502 for the word "b" and leaf nodes 504 and 506. For one embodiment, leaf node 504 represents a "Yes" cluster having triphones ba, be, bi, and etc. and leaf node 506 represents a "No" cluster having triphones bo, bu, and etc. which was generated by a rule-based approach. The triphones ba, be, and bi of leaf node 504 may be moved to leaf node 506 based on a maximum likelihood as explained in **FIG. 4**.

FIG. 6 is an exemplary diagram of a maximum likelihood decision tree 600 to illustrate decoding using the maximum likelihood decision tree. **FIG. 6** shows how to find the appropriate triphone class (i.e., leaf node) via a ML decision tree during decoding or speech recognition.

Referring to FIG. 6, the decision tree 600 includes a plurality of nodes 602 through 618. Each of the nodes 602 through 618 correspond to one class. There is node (i.e., node 602), which contains all the known allophones of this class. For example, node 602 includes the known allophones (t1, t2, t3, t4, t5, t6, t7, and t8), which is also the sum of the allophones for its two child or "yes-no" nodes 604 and 606. Node 604 relates to the allophones of (t1, t2, t3, t4, t5) and node 606 relates to the allophones (t6, t7, t8). Node 608 relates to the allophones (t1, t2, t3), leaf node 616 relates to the allophone (t1), and leaf node 618 relates to the allophone (t2, t3). Also, leaf node 610 relates to the allophones (t4, t5), leaf node 612 relates to the allophone (t6, t7), and leaf node 614 relates to the allophone (t8).

During a speech recognition or decoding process, each triphone of an unknown speech signal must be matched with a suitable class that corresponds to the triphone in the decision tree 600. For example, during the speech recognition or decoding process, the process traverses the decision tree according to the class membership information stored in each tree node, that is, if it belongs to the class of "yes" node, then it goes into "yes" node, otherwise, it goes into the "no" node. This process proceeds on until it reaches a leaf node, where the final triphone class can be found. If this triphone has not been found in the decision tree, it will be dealt with in traditional way of which the phonetic context based leave node class is used.

FIG. 7 is a flow chart illustrating an operation 700 to decode speech using an maximum likelihood (ML) tree according to one embodiment.

At operation 702, all the seen allophones of this class are saved on the ML decision tree root node (from training knowledge). At operation 704, the decision tree is traversed according to the class membership information stored in each tree node, that is, if it belongs to the class of "yes" node, then it goes into "yes" node, otherwise, it goes into the "no" node.

At operation 706, a determination is made if a leaf node has been reached. If a leaf node has been reached, operation 700 continues at operation 718 in which a final decision is determined about a triphone X belonging to which class. If a leaf node has not been reached, operation 700 continues at operation 708.

At operation 708, a determination is made if a triphone X belongs to a Yes class. If it does belong to the Yes class, operation 700 continues at operation 710. At

operation 710, the operation goes into the Yes node and continues at operation 702. If it does not belong to the Yes class, operation 700 continues at operation 712.

At operation 712, a determination is made if triphone X belongs to the No class. If it does belong to the No class, operation 700 continues at operation 714. At operation 714, the operation goes into the No node and continues at operation 702. If it does not belong to the No class, operation 700 continues at operation 716.

At operation 716, the operation selects the phonetic context based leave node and continues at operation 702.

Thus, a method and system for generating and searching an optimal maximum likelihood decision tree for hidden markov model (HMM) based speech recognition have been described. In the foregoing specification, the invention has been described with reference to specific exemplary embodiments thereof. It will, however, be evident that various modifications and changes may be made thereto without departing from the broader spirit and scope of the invention as set forth in the appended claims. The specification and drawings are, accordingly, to be regarded in an illustrative sense rather than a restrictive sense.